Chapter 5

Graphs-Data Representation

5.1 Introduction

Mathematical **Plot**ting **Lib**rary or Matplotlib is a Python library (which is to be installed separately) used for working with graphs, charts, presentations etc. It will work along with NumPy only. Using the modules in Matplotlib we can draw 2D graphs or 3D graphs. It is useful for visualization of data, prediction, trendsetting, comparison etc. It is widely using for scientific computing also.

Matplotlib will convert the data stored using NumPy arrays or Python lists to different types of graphical visualizations. We can save the graph in the computer in any format like jpeg, bmp, png, etc. There are many user-friendly functions in this library for proper visualization. Some of the important functions are listed below. To work with these instructions, we must import *matplotlib* in the programme.

5.2 Preparatory functions

Most of the functions in this category are optional and defined in the module matplotlib.pyplot which is to be imported

1. *title('text'):* This is a function to write the title for the graph.
   Eg:- *title('Path of a freely falling body')*
2. *label('text'):* This is a function to write a label for the x-axis of the graph.
   Eg:- *xlabel('Time')*
3. *ylabel('text'):* This is a function to write a label for the y-axis of the graph.
   Eg:- *ylabel('vertical height')*
4. *legend([' text 1', ' text 2', . . .]):* This is a function to write legends, if different plots are making in a single graph. The computer will automatically assign different colours for different plots. If we want, we can assign the colour. For the first plot, the computer will assign the first string in the list attached to legend() and so on.
   Eg:- If we have 3 plots in a single graph the legend function must be like *legend([' Initial vel =10 m/s', ' Initial vel =20 m/s', ' Initial vel =30 m/s'])*
5. *grid(True):* This is the function to be written if we want a background grid. If we don't want it, avoid this function or write it as *grid(False)*.
6. *figure():* This is to mark different graphs drawing in the same programme is to be visualized with different names in different screens. We can toggle between these screens for comparison. If we have one graph only, this instruction can be

ignored. If we have more than one graph in a single programme, mark as *figure(1)* for the first graph, *figure(2)* for the second graph, so on. If we are not using this function, the computer will draw all the graphs in a single figure.

7. *subplot(m, n, N):* This function is to instruct the computer to show all graphs in a programme on a single screen, but as separate graphs. System will divide the computer screen into rows and columns. Here m stands for the number of rows, n stands for the number of columns and N stands for the graph number.
   Eg:- subplot(2,3,1) will instruct the computer to split the screen into 6 divisions with 2 rows and 3 columns. The number 1 is to instruct the computer to insert the next graph in the first slot. If we want to insert it in the 4th slot, the instruction will be *subplot(2,3,4)*.
8. *show():* The computer will store all types of graphs drawn with different instruction in RAM only. To visualize graph on the screen, we must add this function at the end of the programme.

5.3 Scaling of an axis

Normally the selection of proper scale is by default on both axes. If we need to start from a particular value other than zero (kink in the graph), we can do it with a scaling function. The functions for these in matplotlib are the following.

xlim(lower value, upper value) for x axis and

ylim(lower value, upper value) for y axis. (Read as x-lim and y-lim)

Eg : xlim(60,100)

The same can be done by the following single function.

axis(x-lower limit, x-upper limit, y-lower limit, y-upper limit)

Eg : *axis([0,15,0,2])*

5.4 x-y graph

The function to draw an x-y graph is

plot(data1,data1, color=' ', linestyle = ' ').

The data must be stored in two different arrays in the same sequential order.

Data stored in the array named 'data1' will act as the coordinates for the x-axis and 'data2' will act as coordinates for the Y-axis. Then the system will plot the graph. Colour and line style instructions are optional. If not instructed colour will change cyclically. But line-style will be 'solid' always.

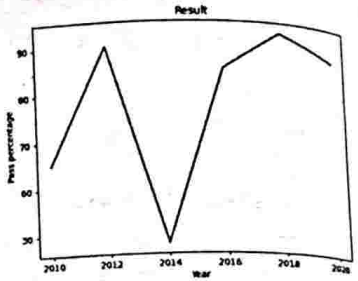
Colour	Code	Line style	Code
blue	b	Solid	-
green	g	Dotted	:
red	r	Dashed	--
cyan	c	Dashdot	-.
magenta	m	None	' or '
yellow	y		
black	k		
white	w		

We can plot any number of lines in the same graph paper (multiple plots) or on the same screen as tiles of graphs. The computer will assign different colours for different graphs and that can be distinguished using the legend function. The scaling of axes is automatic or we can control it by using *axis()*. The attached list of colour code and line style code can be used:

Eg:-*plot(xdta, ydta, color='r', linestyle = ':')*. The value for the x-axis is to be stored in the array 'xdta' and the value for the y-axis in the array 'ydta'.

Example 1:

```
from matplotlib.pyplot import *
from numpy import *
year=[2010, 2012,2014,2016,2018,2020]
res=[65,90,48,85,93,88]
#title('Result')
xlabel('Year')
ylabel('Pass percentage ')
grid(False)
plot(year, res, color='k')
show()
#end
```



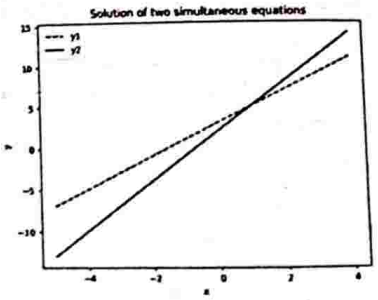
Example 2:

Draw graphs for the following equations and plot it on same paper.

y=2x+3 and y=3x+2

Answer :

```
from matplotlib.pyplot import *
from numpy import *
x=arange(-5,5,1)
y1=2*x+3
y2=3*x+2
plot(x,y1, linestyle='--')
plot(x,y2, linestyle='-')
xlabel('x')
ylabel('y')
grid(False)
```



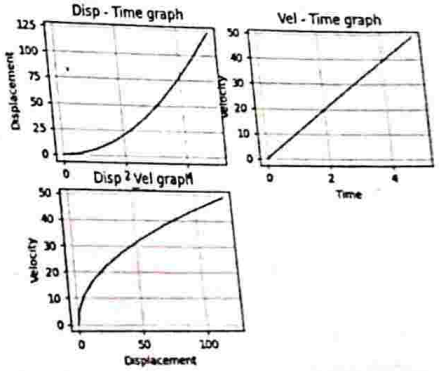

```
title('Solution of two simultaneous equations')
legend(['y1','y2'])
show()
#end
```

Example 3:

Draw s-t graph, v-t graph and s-v graph of a freely falling body on a single screen.

Answer:

```
#freely falling body
from matplotlib.pyplot import *
from numpy import *
tf=input('Enter the final time ')
tf=int(tf)
a=9.8
u=0
t=linspace(0,tf,num=10)
v=u+a*t
s=u*t+0.5*a*t*t
subplot(2,2,1)
title('Disp - Time graph')
xlabel('Time')
ylabel('Displacement')
grid(True)
plot(t,s)
subplot(2,2,2)
xlabel('Time')
ylabel('Velocity')
title('Vel - Time graph')
grid(True)
plot(t,v)
subplot(2,2,3)
xlabel('Displacement')
ylabel('Velocity')
title('Disp - Vel graph')
```



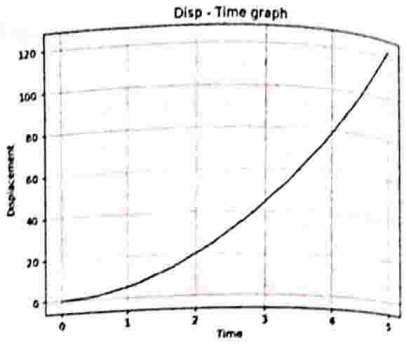
```
grid(True)
plot(s,v)
show()
#end
```

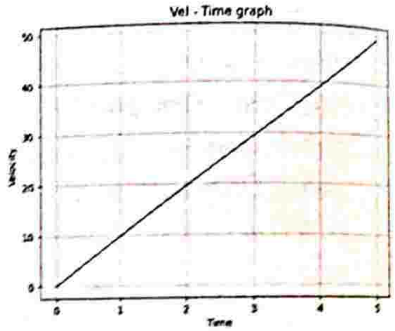
Example 4:

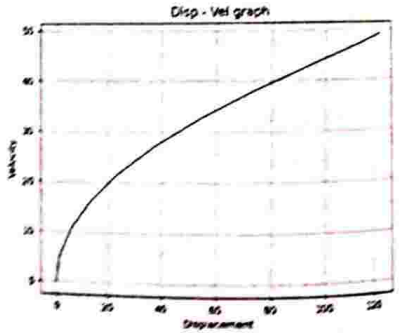
Draw s-t graph, v-t graph and s-v graph of a freely falling body separately using a single programme.

Answer:

```
#freely falling body
from matplotlib.pyplot import *
from numpy import *
tf=input('Enter the final time  ')
tf=int(tf)
a=9.8
u=0
t=linspace(0,tf,num=10)
v=u+a*t
s=u*t+0.5*a*t*t
figure(1)
title('Disp - Time graph')
xlabel('Time')
ylabel('Displacement')
grid(True)
plot(t,s)
figure(2)
xlabel('Time')
ylabel('Velocity')
title('Vel - Time graph')
grid(True)
plot(t,v)
figure(3)
xlabel('Displacement')
ylabel('Velocity')
```








```
title('Disp - Vel graph')
grid(True)
plot(s,v)
show()
#end
```

5.5 Bar Chart

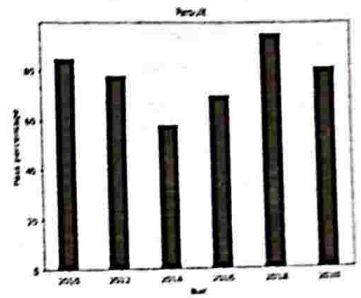
To draw a vertical bar chart using the data stored in two arrays we can use the function bar(Hor. data 1, Vert data 2, color=' '). Colour is optional. The array named data1 will act as the horizontal axis and data 2 will act as the vertical axis.

Example 5:

Draw a bar chart presentation of values stored in two arrays.

Answer:

```
from matplotlib.pyplot import *
from numpy import *
xdata=[2010,2012,2014,2016,2018,2020]
ydata=[85,78,58,70,95,81]
title('Result')
xlabel('Year')
ylabel('Pass percentage')
bar(xdata, ydata, color='k')
show()
#end
```



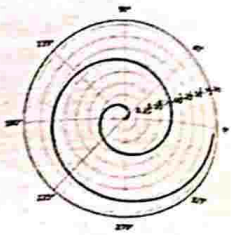
5.6 Polar Plots

It works with mathematical equations in the polar coordinate system. In a polar coordinate system, a point on a two-dimensional plane can be located with one radius and one angle. So, by giving different values of angle and radius as two arrays, we can locate all points in the equation. Locus of these points will mark to form a sketch, named polar plots. The function for this is *polar(theta, radius)*

Example 6:

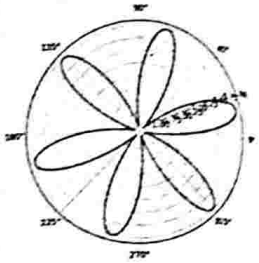
```
from matplotlib.pyplot import *
from numpy import *
r = arange(0, 2, 0.01)
#An array for  radius is created
```

```
theta = 2 * pi * r
#An array for angle is created
polar(theta, r, color='k')
show()
#end
```



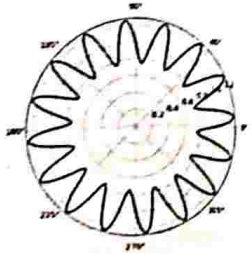
Example 7:

```
from matplotlib.pyplot import *
from numpy import *
th = linspace(0, 2*pi,1000)
r = sin(6*th)
polar(th,r)
show()
#end
```



Example 8:

```
from matplotlib.pyplot import *
from numpy import *
#subplot(polar=True)
th = linspace(0, 2*pi,1000)
r = sin(16*th)*0.25+1
polar(th,r, color='k')
show()
#end
```



5.7 Pie Chart

A pie chart is a type of data visualization that is used to illustrate numerical proportions in data. It is a circular chart in which a circle is divided into sectors according to the percentage value of data items stored in two arrays/lists. One array or list must contain the item labels and the other is for the percentage value. The total of the percentage values must be 100 and both the arrays must have the same length. Pie charts are useful to make a quick comparison.

The function for this is :

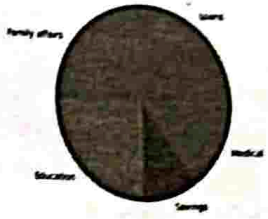
pie(name of list/array of percentages , labels= name of list/array of labels).

Example 9:

```
from matplotlib.pyplot import *
```



```
from numpy import *
lab =["Loans", "Family affairs", "Education",
"Savings", "Medical"]
val=[30, 20, 25, 10, 15]
pie(val, labels=lab)
show()
#end
```



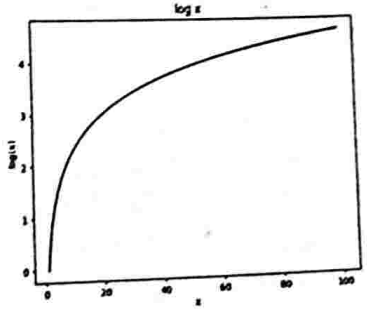
5.8 Mathematical functions

Using the modules in Matplotlib we can draw 2D graphs or 3D graphs of all mathematical functions. Few important mathematical visualizations.

In all cases, we will design a set of x values using the 'linspace' function or 'arrange' function. This list/array is substituted in the equation. If the operand of an equation is a sequence, the result also will be a sequence of the same type. So, we will get an array/list of y values that can be plotted.

Example 10 : log(x)

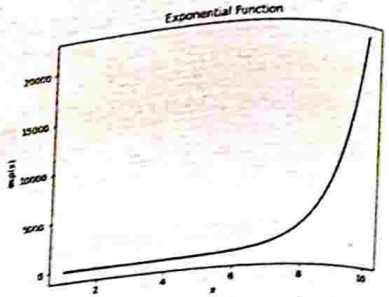
```
#Programme to plot log x
from matplotlib.pyplot import *
from numpy import *
x=linspace(1,100,200)
y=log(x)
xlabel('x')
ylabel('log(x)')
title('log x')
plot(x,y,color='k')
show()
#end
```



Example 11 : exp(x)

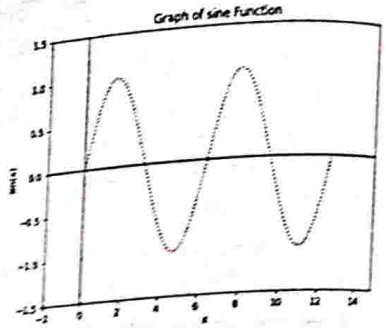
```
#programme to plot exp x
from matplotlib.pyplot import *
from numpy import *
x=linspace(1,10,100)
y=exp(x)
xlabel('x')
```

```
ylabel('exp(x)')
title('Exponential Function')
plot(x,y,color='k')
show()
#end
```



Example 12 : sin(x)

```
#programme to plot sin(x)
from matplotlib.pyplot import *
from numpy import *
#To draw the axis-optional part
xaxis1=[-2,15]
yaxis1=[0,0]
xaxis2=[0,0]
yaxis2=[-1.5,1.5]
plot(xaxis1,yaxis1,color='b')
plot(xaxis2,yaxis2,color='b')
#preperation of graph
x=linspace(0,4*pi,200)
y=sin(x)
xlabel('x')
ylabel('sin(x)')
axis([-2,15,-1.5,1.5])
title('Graph of sine Function')
plot(x,y,color='r',linestyle = ':')
show()
#end
```

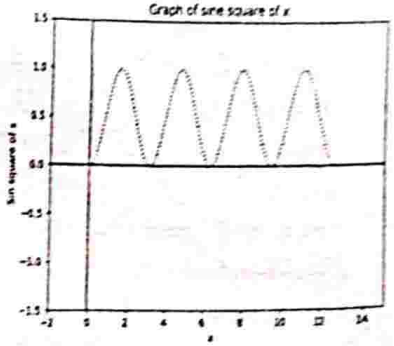


Example 13 : $\sin^2(x)$

```
#programme to plot Sin square of x
from matplotlib.pyplot import *
from numpy import *
xaxis1=[-2,15]
yaxis1=[0,0]
xaxis2=[0,0]
```

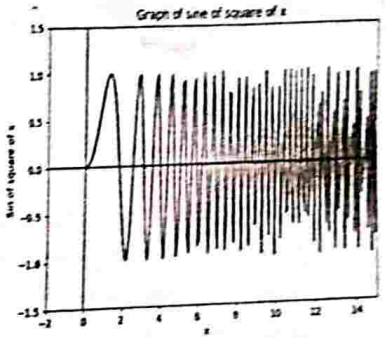


```
yaxis2=[-1.5,1.5]
plot(xaxis1,yaxis1,color='b')
plot(xaxis2,yaxis2,color='b')
x=linspace(0,4*pi,200)
y=sin(x)*sin(x)
xlabel('x')
ylabel('Sin square of x')
axis([-2,15,-1.5,1.5])
title('Graph of sine square of x')
plot(x,y,color='k',linestyle = ':')
show()
#end
```



Example 14: $\sin(x^2)$

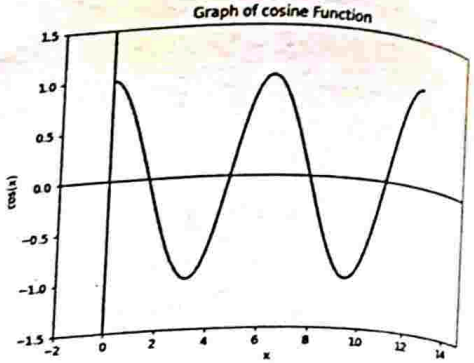
```
#programme to plot Sine of square of x
from matplotlib.pyplot import *
from numpy import *
xaxis1=[-2,15]
yaxis1=[0,0]
xaxis2=[0,0]
yaxis2=[-1.5,1.5]
plot(xaxis1,yaxis1,color='b')
plot(xaxis2,yaxis2,color='b')
x=linspace(0,6*pi,200)
y=sin(x*x)
xlabel('x')
ylabel('Sin of square of x')
axis([-2,15,-1.5,1.5])
title('Graph of sine of square of x')
plot(x,y,color='k')
show()
#end
```



Example 15 : cos(x)

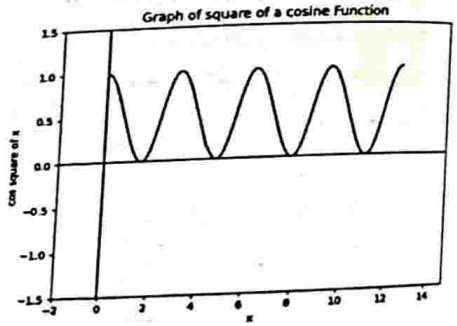
```
#programme to plot cos(x)
```

```
from matplotlib.pyplot import *
from numpy import *
xaxis1=[-2,15]
yaxis1=[0,0]
xaxis2=[0,0]
yaxis2=[-1.5,1.5]
plot(xaxis1,yaxis1,color='b')
plot(xaxis2,yaxis2,color='b')
x=linspace(0,4*pi,200)
y=cos(x)
xlabel('x')
ylabel('cos(x)')
#xlim(0,15)
axis([-2,15,-1.5,1.5])
title('Graph of cosine Function')
plot(x,y,color='k')
show()
#end
```



Example 16 : $\cos^2(x)$

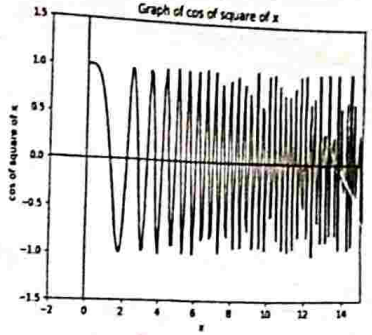
```
#programme to plot square of cos x
from matplotlib.pyplot import *
from numpy import *
xaxis1=[-2,15]
yaxis1=[0,0]
xaxis2=[0,0]
yaxis2=[-1.5,1.5]
plot(xaxis1,yaxis1,color='b')
plot(xaxis2,yaxis2,color='b')
x=linspace(0,4*pi,200)
y=cos(x)*cos(x)
xlabel('x')
ylabel('cos square of x')
#xlim(0,15)
```




```
axis([-2,15,-1.5,1.5])
title('Graph of square of a cosine Function')
plot(x,y,color='k')
show()
#end
```

Example 17 : $\cos(x^2)$

```
#programme for cos of square of x
from matplotlib.pyplot import *
from numpy import *
xaxis1=[-2,15]
yaxis1=[0,0]
xaxis2=[0,0]
yaxis2=[-1.5,1.5]
plot(xaxis1,yaxis1,color='b')
plot(xaxis2,yaxis2,color='b')
x=linspace(0,6*pi,200)
y=cos(x*x)
xlabel('x')
ylabel('cos of square of x')
axis([-2,15,-1.5,1.5])
title('Graph of cos of square of x')
plot(x,y,color='k')
show()
#end
```



Exercise

One-word type questions

1. Name the function which is compulsory in all graphic applications using matplotlib.
2. Name the single function using instead of *xlim* and *ylim*.
3. Name the function in matplotlib to draw a horizontal bar chart.
4. To create 2D graphs we must incorporate . . . . . module of Matplotlib.
5. For data visualization in Python . . . . . library is to be activated.

6. To make a pie chart with Matplotlib, we can use the . . . . . function.
7. Matplotlib will work along with . . . . . module only.

Short answer type questions

1. Write a Python program to draw a straight line with a title and with a suitable label for the x-axis, y-axis.
2. What Is a Python Matplotlib? For what it is used?
3. Read the following code of Python and add the missing lines.

```
import matplotlib.pyplot as plt
plt.plot(x,y)
plt.show()
```

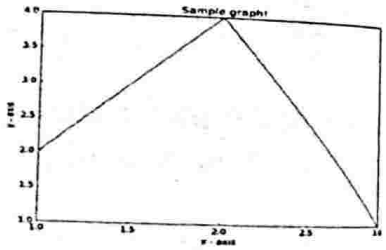
4. Write the functions with proper examples using for writing a title and the names of the x-axis & y-axis of the Python graph.
5. What is the difference between the legend function and grid function?
6. Explain the difference between *plot()* and *subplot()*

Paragraph type questions/Programmes

1. Write a python programme to illustrate the polar chart.
2. Write a python programme to illustrate the bar chart.
3. Write a Python programming to create a pie chart of the popularity of programming languages.

languages:	Java	Python	PHP	JavaScript	C#	C++
Popularity:	22.2	17.6	8.8	8	7.7	6.7

4. Write a short note about the uses and method of using the figure function and subplot function. What is the major difference between them?
5. Observe the attached graph. Write a python programme to get this output with all titles. The plot must be in blue colour.



6. Write a programme to visualize the meeting point of two curves with different equations?
7. Write a python programme to illustrate the pi-chart.
8. Make a mathematical plot of $y=2x^3+5x^2-3x-1$

Long answer type questions

1.a) To draw 3 line-charts of the following data of XY Labs. All the lines must be on the same graph paper with different colours – blue, red, and green. It must be with proper title and legends.

b) Rewrite the programme to display it in 3 separate graph papers.

Density variation at different temperature			
Sample Number	T= 173K	T=273K	T=373K
S 1	774.25	776.06	769.57
S 2	776.03	778.71	781.89
S 3	779.30	782.07	785.65
S 4	779.78	775.53	773.85
S 5	779.65	779.60	779.75

2. Write the Python programme to make the following graphs using a subplot function with proper titles.
   a) x v/s tan x
   b) x v/s sec(x)
   c) x v/s cosec(x)
   d) x v/s cot(x)
3 Write a Python programme to make the following graphs in separate screens using a single programme.
   a) x v/s tan x
   b) x v/s sec(x)
   c) x v/s cosec(x)
   d) x v/s cot(x)

Chapter 4
Numerical Python

4.1 Introduction

Numerical **P**ython or NumPy is a Python library (which is to be installed separately) used for working with arrays and matrix for scientific computing. It includes the functions for linear algebra, Fourier transform, and matrices. In core python, we have lists for manipulating data. But it is a slow process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray. NumPy library provides a lot of supporting functions that make working with data easier. When coming to science-Physics and Maths mainly - this can be easily converted as Matrix or vectors (In python array and matrix are different in properties and usage. So, using the built-in functions of the module NumPy, we can make many matrix and vector operations easily which will lead to many other parts of Mathematical physics. To draw graphs also, we need arrays to store data.

An array is a table of elements of the same data type. Each element is indexed by positive integers. An array may be one dimensional like a list or multidimensional like a spreadsheet, matrix etc. The term multidimensional refers to arrays having several dimensions or axes. The number of axes or dimension is called a rank.

A= [3] is a zero-dimensional array representing a scalar

A= [1, 2, 4] is a one-dimensional array like a row matrix or a vector. It is an array of rank 1.

$A = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix}$ is a two-dimensional array with rows of length 2 and columns of length 3. It is identical to 2X3 Matrix. It is an array of rank 2. It is equivalent to

A = [(1,0,1), (2,1,0)]

Similarly, we can make arrays of any rank.

4.2 Array Creation

To activate NumPy we can use import instruction:

import numpy or

import numpy as np or

*from numpy import *.*

We can create an array by using the *array()* function.

ar= array([1,0,1,0,1,2,8])

Now we created an array named 'ar' of one dimension. To create a two-dimensional array, we can use the instruction,

ar= array([(1,0,2),(2,1,3)])

Care should be given to use different types of brackets properly - function bracket, then array bracket, and then bracket for list of elements in one row.

Similarly, we can make arrays of any dimension.

There are many ways to create arrays. For example, we can convert any Python sequences like list, tuple etc. by using the array function. A list can be converted into an array as follows. If a and b are two existing lists, the following instructions will convert it into an array

arr=array(a) - will convert list a into an array arr

array(a+b) will make an array containing all the elements of both lists.

array([a,b]) will create a two-dimensional array with elements of 'a' as the first row and elements of 'b' as the second row

Just like the function *range(x_s, x_f, dx)* to create a sequence (list), *arange(x_s, x_f, dx)* will make an array with numbers starting from x_s up to x_f with increments dx.

ar= arange(0,1,0.2)

System will create an array named 'ar' with elements [0, 0.2, 0.4, 0.6, 0.8]. The following instructions can be used to create different types of arrays. Instead of giving increment, we can instruct the system to make an array with a definite number of elements using the *linspace*(x_s, x_f, num=n) function. n stands for the number of elements in the array.

ar= linspace(1,2,num=25)

Creation of different types of Arrays

1 An empty array (known as place holder array) can be created by the instruction.

empty (shape, dtype= ...).

It will return an uninitialized array of data type, dtype, and given shape(size). The data type is optional. The default value of the data type is float.

ar=empty ((2,3))

It will create a 2X3 array, without data.

ar=empty ((2,3), dtype=float)

Similarly, we can use any data type.

2 An array of zeros with specified shape and type can be created by the instruction *zeros(shape, dtype=. . .)*

ar=zeros((2,3))

ar=zeros((2,3), dtype=int)

3 An array full of 1 with specified shape and type can be created by the instruction *ones(shape, dtype= . . .)*

ar=ones((2,3))

4 An identity matrix can be created using the function *identity(size, dtype=...)*

ar= identity(3,dtype=int)

5 The function *random.rand(shape)* creates an array with random numbers between 0 and 1

ar=random.rand(3,4)

4.3 Array Attributes

Let there is an array named 'arr' in memory. Then the following instructions will work

Instruction	Result in return	Return value
x=len(arr)	Returns the number of rows of an array	x=2
x=arr.size	Returns the total number of elements in the array	x=12
x=arr.shape	Returns the shape of the array. That means the number of rows and columns	x= (2, 3)
x=arr.ndim	Returns the rank of array	x=2
x=arr.dtype	Returns the type of elements	x=dtype('int32')
x=arr.itemsize	Returns the actual size of each date in bytes	x=4

4.4 Input to an array

Data can be input into an array using a loop either for loop or while loop as discussed in the previous chapter. First, we have to construct an empty array with the required empty space using *empty()*. Then use the loop to input data. Go through the following sample. Similarly, you can write programmes in many methods.

Example: 1

```
# Programme to input into a one-dimensional array
from numpy import *
n=input('Enter the Number of data to input ')
```

```
n=int(n)
arr=empty(n, dtype=int)
print("Enter the data, one in one line")
for i in range(0,n,1):
  arr[i] = input()
print('The entered one-dimensional array is ', arr)
print('Thank you')
#end
```

Example 2:

We can input into a two-dimensional array by two nested loop:

```
#programme to input into a two dimensional array
from numpy import *
r=input('Enter the Number of rows :')
r=int(r)
c=input('Enter the Number of columns :')
c=int(c)
arr=empty((r,c), dtype=int)
print("Enter the data, one in one line")
for i in range(0,r,1):
  for j in range(0,c,1):
    arr[i,j] = input()
print('The entered one dimensional array is ',)
print(arr)
print('Thank you')
#end
```

4.5 Indexing and slicing of an array

In an array each element is characterized by a number called index. First element is with index 0, Next element is with index 1. so on. By using index, we can select any element from an array or slice the array like the list. See the following example.

```
>>> a = array([1, 3, 5, 7, 9])  ↵
>>> a[1]  ↵
3                    Element with index 1
>>> a[1:3]  ↵
```



```
array([3, 5])        Element with index 1 to 3
>>> a[3:]  ↵
array([7, 9])        Element with index 3 up to end
>>> a[:3]  ↵
array([1, 3, 5])          From starting up to element with index 3
```

For multidimensional arrays, the behavior is slightly different.

```
>>> a = array([[1, 3, 2],  [4, 5, 6],[7,8,9],[10,11,12]])  ↵
>>> a  ↵
array([[ 1,  3,  2],
  [ 4,  5,  6],
  [ 7,  8,  9],
  [10, 11, 12]])
>>> a[2]  ↵
array([7, 8, 9])
>>> a[1:3]  ↵
array([[4, 5, 6],
     [7, 8, 9]])
>>> a[:2]  ↵
array([[1, 3, 2],
     [4, 5, 6]])
>>> a[0:2]  ↵
```

But we can use index to specify row and column as follows.

```
>>> a[1,2]  ↵
6
```

Row index is 1 and column index is 2. That means second row, third element.

4.6 Printing of array

If we are using the simple print statement, system will disoplay the array in a built-in fashion. One dimensional arrays are printed as rows.

```
>>> arr= arange(6)  ↵
>>> print(arr)  ↵
[0 1 2 3 4 5]
```

Two dimensional arrays are printed as matrices and three dimensional as lists of matrices.

```
>>> arr= array([(1,0,2),(2,1,3)])
>>> print(arr)
[[1 0 2]
 [2 1 3]]
```

This is the default printing. By using loops, we can make printing as we desired.

Example 3:

Write a programme to read a two-dimensional array and print it like a matrix.

Answer:

```
from numpy import *
r=input('Enter the Number of rows :')
r=int(r)
c=input('Enter the Number of columns :')
c=int(c)
arr=empty((r,c), dtype=int)
print("Enter the data, one in one line")
for i in range(0,r,1):
   for j in range(0,c,1):
      arr[i,j] = input()
for i in range (r):
   for j in range(c):
      print(arr[i,j], end=' ')
   print ('\n')
print('Thank you')
#end
```

Example 4:

Write a programme to read a one-dimensional array of 10 elements from keyboard and split into two arrays of 5 elements each.

Answer:

```
#spliting of array
from numpy import *
n=10
#data input
arr=empty((n), dtype=int)
print("Enter the data, one in one line")
for i in range(0,n,1):
   arr[i] = input()
x=int(n/2)
ans1=arr[:x]
ans2=arr[x:]
print(ans1)
print(ans2)
print('Thank you')
#end
```


4.7 Array Modification instructions

An array can be changed with various commands. Go through some important instructions. For full list refer python.org. Go through the following instructions.

resize(m, n) : This is to change the shape of a particular array permanently and store with same identifier. The total number of elements must be compatible.

```
>>> a=array([[1, 2, 3, 4],[5, 6, 7, 8]])
>>> print(a)
[[1 2 3 4]
 [5 6 7 8]]                    It is a 2 X 4 array
>>> a.resize(4,2)
>>> print(a)
[[1 2]
 [3 4]
 [5 6]
 [7 8]]                        It is a 4 X 2 array
```

reshape(m,n) : This is to change the shape of array. That means we can change number of rows and columns. The total number of elements must be compatible. On execution, the original array will not change.

```
>>> a=array([[1, 2, 3, 4],[5, 6, 7, 8]])
>>> print(a)
[[1 2 3 4]
 [5 6 7 8]]
>>> x=a.reshape(4,2)
```

```
>>> print(x)
[[1 2]
 [3 4]
 [5 6]
 [7 8]]
```

ravel() : This is to change a multidimensional array to the format of a linear array and store in an another variable. On execution, the original array will not change. The word ravel means removing the knots.

```
>>> a= array([[1,0,2],[2,1,3]])
>>> print(a)
[[1 0 2]
 [2 1 3]]
```

It is a 2 X 3 array

```
>>> x=a.ravel()
>>> x
array([1, 0, 2, 2, 1, 3])
```

Here 'a' is a multidimensional array, and x is a linear array.

tolist() : This is to convert a one-dimensional array back to list.

```
>>> a=array([1, 0, 2, 2, 1, 3])
>>> print(a)
[1 0 2 2 1 3]
>>> lst=a.tolist()
>>> print(lst)
[1, 0, 2, 2, 1, 3]
```

transpose() : This is to find the transpose of an array

```
>>> a=array([[1, 2, 3, 4],[5, 6, 7, 8]])
>>> print(a)
[[1 2 3 4]
 [5 6 7 8]]
>>> x=a.transpose()
>>> print(x)
[[1 5]
 [2 6]
 [3 7]
 [4 8]]
```


4.8 Arithmetic operations of an array

Basic arithmetic operators will work on arrays. Result can be stored in a new array. Remember that the operation is element wise, and it is not the matrix operations like addition or multiplication of matrix.

```
>>> a=array([[10,20,30],[30,20,10]])
>>> b=array([[20,20,30],[30,10,10]])
>>> c=a+b
>>> d=a-b
>>> e= a*b
>>> f=a/b
>>> print(a)
[[10 20 30]
 [30 20 10]]
>>> print(b)
[[20 20 30]
 [30 10 10]]
>>> print(c)
[[30 40 60]
 [60 30 20]]
>>> print(d)
[[-10   0   0]
 [  0  10   0]]
>>> print(e)
([[200, 400, 900],
 [900, 200, 100]]
>>> print(f)
[[0.5 1.  1. ]
 [1.  2.  1. ]]
```

Instead of the instruction *a*b*, function *multiply(a,b)* also can be used for elementwise multiplication. It is not matrix multiplication. Similarly, instead of the

instruction *a/b*, we can use the function *divide(a,b)* for elementwise division.

We can operate on every element with a single element from outside like the following example.

```
>>> x=a+2 ↵
>>> y=a*10 ↵
>>> print(x) ↵
[[12 22 32]
[32 22 12]]
>>> print(y) ↵
[[100 200 300]
[300 200 100]]
```

We can operate on selected elements also to modify the array.

```
>>> a=array([[10,20,30],[30,20,10]]) ↵
>>> a[1,2]=a[1,2]+5 ↵
>>> print(a) ↵
[[10 20 30] ↵
[30 20 15]] ↵
```

4.9 Matrix

NumPy **matrices** are strictly 2-dimensional, while NumPy **arrays** (ndarray) are N-dimensional. Only matrix format will do matrix operations like addition, multiplication, inverse etc. For that, the data is to be entered in the matrix format.

Creation of Matrix

```
>>> m= mat([[1, 2, 3, 4],[5, 6, 7, 8]]) ↵
>>> m ↵
matrix([[1, 2, 3, 4],
    [5, 6, 7, 8]])
```

We can convert an array to matrix as follows.

```
>>> a=array([[1, 2, 3, 4],[5, 6, 7, 8]]) ↵
>>> m=mat(a) ↵
>>> m ↵
matrix([[1, 2, 3, 4],
        [5, 6, 7, 8]])
```

Similarly, a matrix can be changed to an array as follows.


```
>>> m= mat([[1, 2, 3, 4],[5, 6, 7, 8]]) ↵
>>> a=array(m) ↵
>>> a ↵
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
```

Both conversions are necessary since arithmetic operations on array and matrix are with different scientific sense and application.

4.10 Arithmetic operations of a matrix

Matrix objects are a subclass of ndarray, so they have all the attributes and methods of arrays. Apart from that matrix can perform matrix operations also if the size and shape of both metrices are compatible. Python versions wise there are some slight differences in the instructions. Note that there are many major differences when shifting from version 2.x to version 3.x.

1. Matrix Multiplication

```
>>> a= mat([[1, 2, 3],[4, 5, 6]]) ↵
>>> b=mat([[11, 12],[13,14],[15,16]]) ↵
>>> d=dot(a,b) ↵
>>> e=a@b ↵
>>> a ↵
matrix([[1, 2, 3],
        [4, 5, 6]])
>>> b ↵
matrix([[11, 12],
        [13, 14],
        [15, 16]])
>>> d ↵
matrix([[ 82,  88],
        [199, 214]])
>>> e ↵
matrix([[ 82,  88],
        [199, 214]])
```

Note: functions *dot(a,b) or a@b* depends on the version of Python. Both will give same result. Function *multiply(a,b)* is not a matrix multiplication. It is an element-wise multiplication.

2. Transpose of a matrix

```
>>> a= mat([[1, 2, 3],[4, 5, 6]]) ↵
>>> b=a.T ↵
>>> print(a) ↵
[[1 2 3]
 [4 5 6]]
>>> print(b) ↵
[[1 4]
 [2 5]
 [3 6]]
```

3. Cross product

```
>>> a= mat([[1, 2, 3],[4, 5, 6]]) ↵
>>> b= mat([[11, 12,13],[14,15,16]]) ↵
>>> c=cross(a,b) ↵
>>> a ↵
matrix([[1, 2, 3],
        [4, 5, 6]])
>>> b ↵
matrix([[11, 12, 13],
        [14, 15, 16]])
>>> c ↵
array([[-10,  20, -10],
       [-10,  20, -10]])
```

4. Inverse of Matrix

```
>>> a=mat([[1,1,1],[0,2,5],[2,5,-1]])
>>> b = mat(linalg.inv(a))
>>> a ↵
matrix([[ 1,  1,  1],
        [ 0,  2,  5],
        [ 2,  5, -1]])
>>> b ↵
matrix([[ 1.2857, -0.2857, -0.14285],
```



```
        [-0.4761,  0.1428,  0.2380],
        [ 0.1904,  0.1428, -0.0952 ]])
```

5. Inner Product

```
>>> a=mat([[1,2],[3,4]])
>>> b=mat([[11,12],[13,14]])
>>> c=inner(a,b)
>>> a ↵
matrix([[1, 2],
        [3, 4]])
>>> b ↵
matrix([[11, 12],
        [13, 14]])
>>> c ↵
matrix([[35, 41],
        [81, 95]])
```

6. Trace

```
>>> a= mat([[1, 2, 3],[4, 5, 6],[7, 8, 9]]) ↵
>>> b=trace(a) ↵
>>> a ↵
matrix([[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]])
>>> b ↵
15
```

Example 5:

Write a programme to read two-dimensional matrices from keyboard and change its shape as per the requirement of user.

Answer:

```
from numpy import *
#Initialization
r=input('Enter the Number of rows :')
r=int(r)
c=input('Enter the Number of columns :')
```

```
c=int(c)
arr=empty((r,c), dtype=int)
#data entry
print("Enter the data, one in one line")
for i in range(0,r,1):
  for j in range(0,c,1):
    arr[i,j] = input()
x=input('Enter the new number of rows :')
x=int(x)
y=input('Enter the new number of columns :')
y=int(y)
if r*c!=x*y:
  print('The entered values are not compatible')
  print('Task cannot be completed')
else:
  ans=arr.reshape(x,y)
  #answer printing
  for i in range (x):
    for j in range(y):
      print(ans[i,j], end='    ')
    print ('\n')
#end
```

Example 6:

Write a programme to show the difference between elementwise multiplication and matrix multiplication.

Answer:

```
from numpy import *
#Initialization
print('This programme is applicable only for square matrix')
r=input('Enter the Number of rows of first Matrix :')
r=int(r)
ar1=empty((r,r), dtype=int)
ar2=empty((r,r), dtype=int)
```



```
#data entry
print("Enter the data for first matrix, one in one line")
for i in range(0,r,1):
  for j in range(0,r,1):
    ar1[i,j] = input()
print("Enter the data for second matrix, one in one line")
for i in range(0,r,1):
  for j in range(0,r,1):
    ar2[i,j] = input()
print('Elementwise multiplication')
ans1=ar1*ar2
for i in range (r):
  for j in range(r):
    print(ans1[i,j], end='    ')
  print ('\n')
print('Matrix multiplication')
ans2=ar1@ar2
for i in range (r):
  for j in range(r):
    print(ans2[i,j], end='    ')
  print ('\n')
print('Thank you')
#end
```

Exercise

One-word type questions

1. If in python What is a rank in numerical python?
2. What is the difference between list and array regarding the speed?
3. Name the module which we must import to core Python.
4. Name the instruction to find the rank of an array.
5. A matrix is a multidimensional array of rank ? - - - - -
6. Metrix operations are defined in the- - - - - - module of Python.
7. In matrix each element can be located by its - - - - - - -

8. A= [3] is an example for - - 0 - - dimensional array
9. A= [1, 2, 4] is an example for - - 1 - - dimensional array
10. $A = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix}$ is an example for - 2 - - - dimensional array
11. A = [(1,0,1), (2,1,0)] is an example for - - 2 - - dimensional array.
12. A = [(1,0,1), (2,1,0)] is an example for array of rank - - - - - - -
13. *ravel()* is a function for - - - - - - -

Short answer type questions

1. What is the core difference between the instructions a*b and a@b?
2. Write a programme segment to read data for a two-dimensional array using the loop.
3. Write a programme segment to print a two-dimensional array using the loop.
4. What is the difference between the shape function and ndim function in NumPy?
5. What is the difference between the size and shape of an array in Python
6. Write 5 array attributes of Python arrays.
7. Write program segments to create 1. Empty array 2. Array filled with zeros.
8. Write 3 methods to create a matrix.

Paragraph type questions/Programmes

21. Explain the concept of an array in NumPy with explanations for Rank and Dimension.
22. Write a full programme to read a matrix and to print the matrix and its transpose in matrix format using loops.
23. Write a full programme to read two matrices from the keyboard and to print its product using loops.
24. Write a Python programme to read a Matrix and to print its transpose and inverse.
25. Write a programme to create an empty matrix for a requested shape and to fill it with data. The programme must give a printout of the data in matrix format.
26. What is the fundamental difference between resize() and reshape(). Write a programme segment to demonstrate it.
27. Write a program segment to demonstrate the use of slicing of an array with a numerical example.

Long answer type questions

1. Explain the following operations on a matrix with examples and codes of Python.
   1. Multiplication, 2. Transpose, 3. Trace,
   4. inverse of a matrix, 5. Inner product, 6. Cross product
2. Explain different methods of creation of an array and mathematical operations of the array with Python code and example.
3. Write a program to read the list, then convert it into an array and to print the array using loops.